Service-Based 3D Rendering and Interactive 3D Visualization

Benjamin Hagedorn
Hasso-Plattner-Institut
Prof.-Dr.-Helmert-Str. 2-3
D-14482 Potsdam
benjamin.hagedorn@hpi.uni-potsdam.de

Jürgen Döllner
Hasso-Plattner-Institut
Prof.-Dr.-Helmert-Str. 2-3
D-14482 Potsdam
doellner@hpi.uni-potsdam.de

Abstract

This report describes the subject and preliminary results of our work in the context of the HPI Future SOC Lab, which generally aims on how to exploit high performance computing (HPC) capabilities for service-based 3D rendering and service-based, interactive 3D visualization. A major focus is on the application of HPC technologies for the creation, management, analysis, and visualization of and interaction with virtual 3D environments, especially with complex 3D city models.

1 Motivation

Virtual 3D city models represent a major type of virtual 3D environments. They can be defined as a digital, geo-referenced representation of spatial objects, structures and phenomena of a distinct geographical area; its components are specified by geometrical, topological, graphical and semantically data and in different levels of detail.

Virtual 3D city models are, e.g., composed of digital terrain models, aerial images, building models, vegetation models, and city furniture models. In general, virtual 3D city models serve as information models that can be used for 3D presentation, 3D analysis, and 3D simulation. Today, virtual 3D city models are used, e.g., for urban planning, mobile network planning, noise pollution mapping, disaster management, or 3D car and pedestrian navigation.

In general, virtual 3D city models represent prominent media for the communication of complex spatial data and situations, as they seamlessly integrate heterogeneous spatial information in a common reference frame and also serve as an innovative, effective user interface. Based on this, virtual 3D city models, as integration platforms for spatial information, represent essential building blocks of today's and future information infrastructures.

1.1 Complexity of 3D city models

Virtual 3D city models are inherently complex in multiple dimensions, e.g., semantics, geometry, appearance, and storage. Major complexities are described in the following:

Massive amounts of data: Virtual 3D city models typically include massive amounts of image data (e.g., aerial images and façade images) as well as massive amounts of geometry data (e.g., large number of simple building models, or smaller number of buildings modeled in high detail). Vegetation models represent another source of massive data size; a single tree model could contain, e.g., approximately 150,000 polygons).

Distributed resources: In today's so called geospatial data infrastructures (GDIs), the different components (i.e., base data) of virtual 3D city models as well as functionalities to access, and process (e.g., analyze) virtual 3D city models can be distributed over the Internet. In specific use cases such as in emergency response scenarios, they need to be identified, assembled, and accessed in an ad-hoc manner.

Heterogeneity: Virtual 3D city models are inherently heterogeneous, e.g., in syntax (file formats), schemas (description models), and semantics (conceptual models).

As an example, the virtual 3D city model of Berlin contains about 550,000 building models in moderate and/or high detail, textured with more than 3 million single (real-world) façade textures. The aerial image of Berlin (covering an area of around 850 km2) has a data size of 250 GB. Together with additional thematic data (public transport data, land value data, solar potential) the total size of the virtual 3D city model of Berlin is about 700 GB.

1.2 Service-based approach

The various complexities of virtual 3D city models have an impact on their creation, analysis, publishing, and usage. Our overall approach to tackle these complexities and to cope with these challenges is to design and develop a distributed 3D geovisualization system as a technical framework for 3D geodata integration, analysis, and usage. For this, we apply and combine principles from Service-Oriented Computing (SOC), general principles from 3D visualization systems, and standards of the Open Geospatial Consortium (OGC).

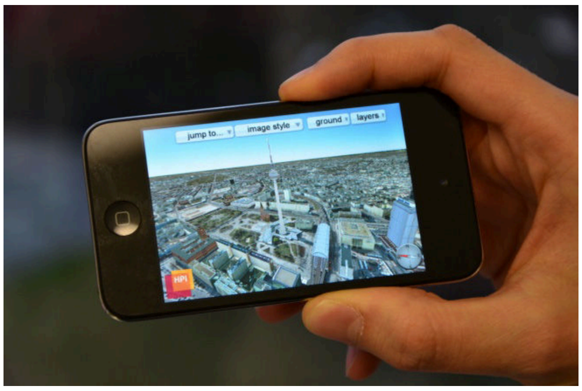

Figure 1: 3D client for exploring the 3D city model of Berlin, running on an iPod.

To make complex 3D city models available even for small devices (e.g., smart phones, tablets), we have developed a client/server-system that is based on server-side management and 3D rendering [1]: A portrayal server is hosting a 3D city model in a preprocessed form that is optimized for rendering, synthesizes images of 3D views of this data, and transfers these images to a client, which (in the simplest case) only displays these images. By this, the 3D client is decoupled from the complexity of the underlying 3D geodata. Also, we can optimize data structures, algorithms and rendering techniques with respect to specialized software and hardware for 3D geodata management and 3D rendering at the server-side. – Figure 1 shows our 3D client running on an iPod; it allows a user to interactively explore the virtual 3D city model of Berlin.

Our project in the context of the HPI Future SOC Lab aims on research and development of how to exploit its capabilities for such a distributed 3D visualization system, especially for 3D geodata preprocessing, analysis, and visualization. The capabilities of interest include the availability of many cores, large main-memory, GPGPU-based computing, and parallel rendering.

2 Processing massive 3D city models

As raw geodata cannot be used directly for visualization and rendering purposes, this data needs to be transformed into graphics representations that can be rendered by a 3D rendering engine. Geodata preparation includes preprocessing of terrain data (geometry and textures, e.g., aerial images) as well as preprocessing of building data (geometry and façade textures). Building data are, e.g., originally provided in the CityGML format, an XML-based standard model and format for the specification of 3D city models, including semantics, geometry, attributes, and appearance information. Figure 2 illustrates the preprocessing of such CityGML building data, which includes the following three major stages and sub tasks:

Data extraction: building feature extraction, geometry extraction and triangulation, object-id assignment to each building object, spatial organization of all buildings in a quadtree structure.

Geometry optimization: geometry batching, data serialization.

Façade texture optimization: texture atlas computation, texture atlas tiling, texture coordinate adjustment.

Typically, the texture data of a large virtual 3D city model does not completely fit into a graphics card's texture memory. Thus, rendering large 3D city models requires selecting and loading the data (in appropriate level of detail) that is required for a specific camera position and view frustum. Texture preprocessing and optimization is a time-consuming stage

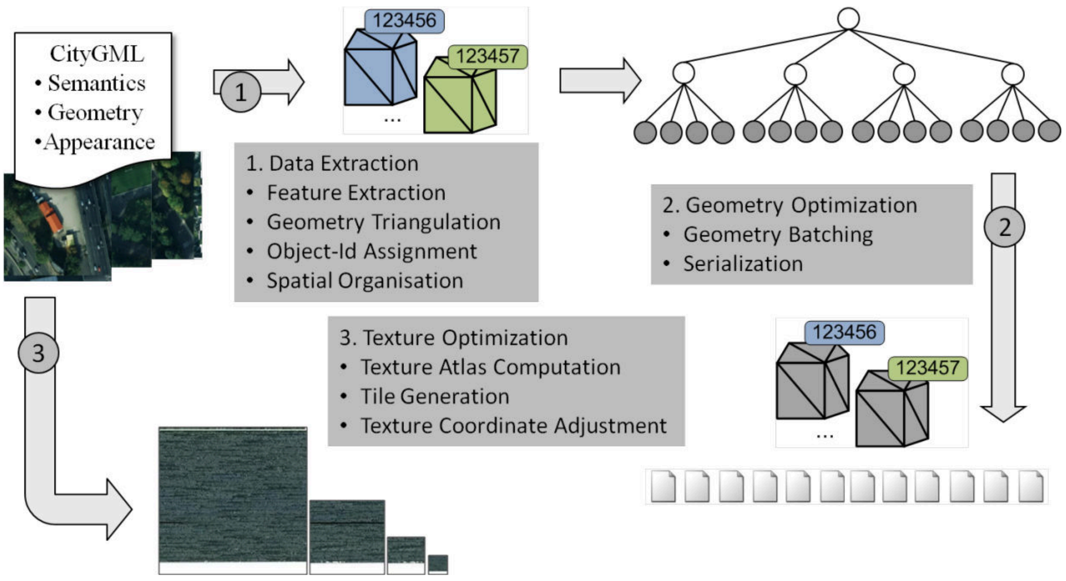

Figure 2: Preprocessing scheme for 3D geodata for 3D rendering.

in the preparation of massive 3D building data for efficient rendering and visualization. It includes a) arranging many single façade textures in texture atlases which represent parts of a very large virtual texture and b) cutting these texture atlases into even smaller parts that could be selected and handed over to the graphics card to render a specific view.

Our original implementation of this texture preprocessing stage was designed for being executed on standard desktop PCs. Due to relatively small main memory and disk space (compared to HPC servers), it had to encode and store intermediate texture data as image files on the hard disk. Starting with this implementation we experimented to take advantage of the HPC servers of the HPI Future SOC Lab to reduce the time for texture preprocessing.

First experiments included to reduce I/O-related overhead of this process. For this, we set up and used a 160 GB large virtual RAM drive in the server's main memory. Only through this, we extremely reduced the time to preprocess the massive 3D city model of Berlin (ca. 550.000 buildings including façade textures) from more than a week on a desktop PC (2.80 GHz, 8 logical cores; 6 GB main memory) to less than 15 hours on the Future SOC Lab's RX600-S5-1 (256 GB RAM; 48 logical cores).

Also, we redesigned the implementation of our preprocessing tools to take advantage of potentially very large main memory (for storing intermediate texture data instead of encoding it and writing to hard disk) and large number of available threads (for increasing the degree of parallel tiling of texture atlases).

3 Processing massive 3D point clouds

3D point clouds are another major source of 3D geodata, which are collected, e.g., via airborne or terrestrial laser scanning. 3D point clouds can represent a digital surface model of the earth's surface and are a starting point for deriving high-level data, e.g., based on classification, filtering, and reconstruction algorithms. For such algorithms it is crucial to be able to handle and manage the often very large 3D point data sets. 3D point clouds of a single city can easily contain several billion points. In the past, we had developed a set of algorithms and tools to cope with this challenge and to process, analyze, and visualize massive 3D point clouds [2, 3].

Spatial organization and rasterization are two major preprocessing tasks for 3D point clouds:

Spatial organization: To efficiently access and spatially analyze 3D points, they need to be ordered in a way that allows efficient access to the data; quadtrees and octrees represent common structures for their organization.

Rasterization: Rasterized 3D point clouds are a central component for visualization techniques and processing algorithms, as they allow efficient access to points within a specific bounding box. Rasterization transforms arbitrary distributed 3D points into a gridded, reduced, and consolidated representation; representative points are selected and missing points are computed and complemented. Rasterized point clouds are used, e.g., for the computation of triangulated surface models, for consistent level-of-detail techniques, and other efficient processing algorithms.

In the context of the Future SOC Lab we have started to research on how the HPC capabilities can help to improve speed and quality of these two tasks.

3.1 Spatially organizing 3D point clouds

To create a quadtree/octree structure, we have used the PARTREE algorithm, a parallel algorithm that creates several trees that are combined to a single one later (Figure 3).

The parallel quadtree/octree generation process for 3D point clouds was implemented based on OpenMP. We tested our implementation with data sets of up to 26.4 million 3D points. In this setting, the HPC system with 8 logical cores (FluiDyna Typhoon, 2.4 GHz, 24 GB RAM, 8 logical cores) was faster than a Desktop PC with 4 logical cores; however, the gain in time is not dramatically (Figure 4). The time required to merge the generated component trees into one tree is increasing with their number (i.e., the number of threads); in this step, the HPC server outperforms the desktop system.

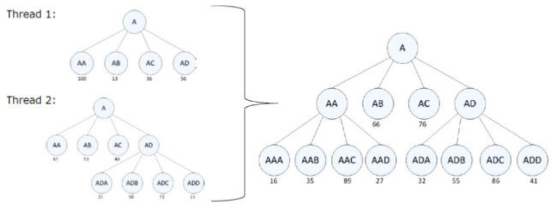

Figure 3: Functional scheme of PARTREE algorithm. Two threads create two local trees (here: quadtrees) that are merged then into a single one.

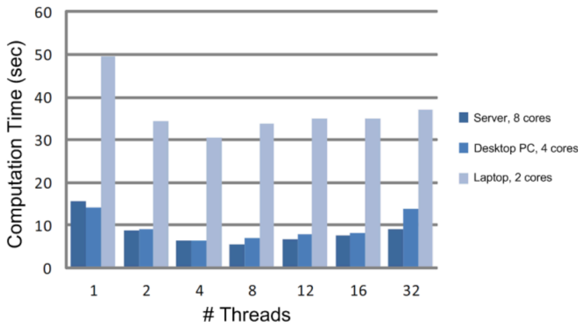

Figure 4: Time to compute an octree from 26.4 million 3D points by PARTREE method.

3.2 Rasterization of 3D point clouds

Rasterization is a multi-step process which 1) identifies for each point of an unordered 3D point cloud the corresponding raster cell and assigns the raster cell ID, 2) ordering the points according to their raster cell ID, 3) computing one characteristic 3D point for each raster cell (representing the relevant input points), and 4) interpolating cell points for empty raster cells. For sorting the points according to their raster cell ID, our implementation uses the Bitonicsort algorithm, which requires to establish a bitonic order before sorting.

We have implemented the rasterization algorithm in two versions: a) a version for multi-core processors (using OpenMP) and b) a data-parallel GPU-based version (using CUDA). The CUDA version has been tested with the Future SOC Lab's TESLA system. It can rasterize a point cloud of 30.5 million 3D points in only 22 minutes in contrast to more than 5 hours of a single threaded CPU-version.

4 Next Steps

The continuation of our work in the area of exploitation of HPC capabilities for service-based 3D rendering and 3D visualization systems and in the context of the HPI Future SOC Lab will include further improvement of our algorithms, processes, and tools. Also, we plan to extend our work to research and development on HPC-based analysis of massive 3D geodata (such as solar potential analysis) and on service-based technologies for assisted interaction and camera control in massive virtual 3D city models [4]. For this, we will exploit parallel and GPU-based algorithms to generate so called "best views" on virtual 3D city models based on visual, geometrical and usage-related characteristics.

5 Conclusions

This report briefly described the subject of our research and development in the context of the HPI future SOC Lab, preliminary results, as well as intended future work. Work and results were mainly in the areas of preprocessing massive 3D city model data and processing of massive 3D point clouds. Here, we could dramatically increase the time required to preprocess raw 3D geodata (CityGML data with geometry and textures; and massive 3D point clouds). Also, we identified additional opportunities for optimizing these algorithms. More generally, this work leads to new opportunities for research and development on advanced and innovative technologies for the exploitation (e.g., analysis and visualization) of massive spatial 3D data sets.

Acknowledgment

We would like to thank Markus Behrens for his help with the implementation and evaluation of 3D point cloud processing and its evaluation.

References

[1] D. Hildebrandt, J. Klimke, B. Hagedorn, J. Döllner: Service-oriented Interactive 3D Visualization of Massive 3D City Models on Thin Clients. *In: Proc. of 2nd Int. Conf. on Computing for Geospatial Research & Application COM.Geo 2011*, 2011.

[2] R. Richter, J. Döllner: Out-of-Core Real-Time Visualization of Massive 3D Point Clouds. *In: Proc. of 7th Int. Conf. on Virtual Reality, Computer Graphics, Visualisation and Interaction in Africa*, pp. 121-128, 2010.

[3] R. Richter, J. Döllner: Potentiale von massiven 3D Punktwolkendatenströmen. *In: Proc. of Geoinformatik 2012*, Braunschweig, 2012.

[4] J. Klimke, B. Hagedorn, J. Döllner: A Service-based Concept for Camera Control in 3D Geovirtual Environments. *In: Proc. of 7th Int. 3D GeoInfo Conference 2012*. (accepted)